Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2007, Article ID 35010, 12 pages
doi:10.1155/2007/35010

Research Article

Embedded Active Vision System Based on an FPGA Architecture

Pierre Chalimbaud and François Berry

Laboratoire des Sciences et Matériaux pour l'Elecronique, et d'Automatique (LASMEA), UMR 6602 du CNRS, Université Blaise-Pascal 24 Avenue des Landais, 63177 Aubiere Cedex, France

Received 2 May 2006; Accepted 14 September 2006

Recommended by Christoph Grimm

In computer vision and more particularly in vision processing, the impressive evolution of algorithms and the emergence of new techniques dramatically increase algorithm complexity. In this paper, a novel FPGA-based architecture dedicated to active vision (and more precisely early vision) is proposed. Active vision appears as an alternative approach to deal with artificial vision problems. The central idea is to take into account the perceptual aspects of visual tasks, inspired by biological vision systems. For this reason, we propose an original approach based on a system on programmable chip implemented in an FPGA connected to a CMOS imager and an inertial set. With such a structure based on reprogrammable devices, this system admits a high degree of versatility and allows the implementation of parallel image processing algorithms.

1. INTRODUCTION

FPDs and in particular FPGAs have achieved rapid acceptance and growth over the past decade because they can be applied to a very wide range of applications [1]. One of the most interesting applications of FPGAs is the prototyping of designs to be implemented as gate arrays. Another is the emulation of entire large hardware systems. Apart from prototyping, an emerging topic for FPGA applications is their use in custom computing machines. This involves using the programmable parts to "execute" software, rather than compiling the software for execution on a regular CPU. For the latter, the notion of such soft-core CPU or hardware overload of the instruction set becomes crucial. Such approaches offer a good tradeoff between the performance of fixed-functionality hardware and the flexibility of software-programmable substrates. These different aspects are a great advantage in the design of an embedded sensing system, in particular when there are several data flows. Like ASICs, the main benefit of these systems is their ability to implement specialized circuitry directly in hardware. However, fast prototyping is easier for FPGAs. Consequently, in the design of a versatile embedded system dedicated to image processing, the FPGA solution proves to be the better way.

In computer vision and especially in vision processing, the impressive evolution of algorithms and the emergence of new techniques drastically increase the complexity of algorithms. This computational aspect is crucial for the majority of real-time applications and in most cases programmable devices are the best option. For example, FPGAs have already been used to accelerate real-time point tracking [2], stereovision computing [3], color-based object detection [4], and video and image compression [5].

In this paper, an architecture dedicated to computer vision is proposed. Our approach towards a smart camera consists in performing most of the early vision processing at the sensor level, before transmitting the information to the main processing unit. This behavior is inspired by the human vision system, where eyes are responsible for attention and fixation tasks, sending to the brain only pertinent information about the observed scene. As a matter of fact, the amount of visual data to be transmitted and analyzed is strongly reduced and communication bottlenecks can be avoided. The adaptation of perceptual aspects from biological vision to artificial systems, which is known as active vision and active perception, is briefly explained in Section 2 as the principal motivation of this work. Consequently, the main originality of this work is to use the concepts developed in active vision and more generally in bio-inspired computer vision in order

to design suitable hardware. In Section 3, the hardware of the smart camera is described. The technological choices are argued according to the objectives given in the previous section. The different modules are fully described and the different data flows are explained. Section 4 presents the core of the FPGA design, in particular the specific modules like the address generation unit or the fixed pattern noise (FPN) correction unit. Finally, we present the results of two image processing algorithms (motion detection and high-speed template tracking implementation).

2. ACTIVE VISION SYSTEMS

One of the numerous objectives in artificial vision research is to build computer systems that analyze images automatically, determining what the computer "sees" or "recognizes" and "understands" from the environment.

In what follows, the problem is to perform the process of interpretation of sensorial data within an environmental model. The first ways of treating the "vision problem" used passive vision and dynamic vision approaches. Passive vision comprises the classical analysis of images. The approach that David Marr explicitly advocated [6], to which many others subscribe, has led to a thriving research field that has been dominant in visual science in recent years. From David Marr, "*Vision is a process that produces from images of the external world a description that is useful to the viewer and not cluttered with irrelevant information.*" David Marr proposes a model of visual processing that begins by identifying the "zero-crossings" (edges) in the image, uses this edge information to provide a crude segmentation of surfaces called the 2D sketch, and finally extracts from this sketch the three-dimensional spatial information. That spatial interpretation is expressed in terms of geometrical primitives such as generalized cylinders or cones, so that the only data which must be explicitly stored are the x, y, z locations, alpha, beta, gamma orientations, and aspect ratios of each of the cylinders and a symbolic code of the relations between them. In this way, the complex scene is reduced to a highly compressed set of meaningful numbers. The problem with this model is that nobody has ever been able to define how such spatial information can be reliably extracted from the scene. Moreover, the visual world contains far too many ambiguities to be handled successfully. Dynamic vision is a complementary approach which corresponds to the study of visual information but in an unbounded sequence of views. This approach introduces time into the image processing, while movement (measured by optical flow) is used in the perception process. Some classical approaches using these strategies revolve around recovering structure from motion.

In contrast to these two approaches, [8–10] have proposed the active vision approach. Active vision techniques are derived from attempts to simulate the human visual system. In human vision, head motion, saccadic eye movement, and the eye's adaptation to lighting variations are important in the perception process. Active vision therefore aims to simulate the power of this adaptation. In other words, active vision is an alternative approach to dealing with artificial vision problems. The central idea, also known as the task-driven paradigm, is to take into account the perceptual aspect of visual tasks. Therefore, instead of a full 3D representation of the observed scene, the system is supposed to extract only the information useful for solving a given problem through a task-driven observation strategy (Figure 1).

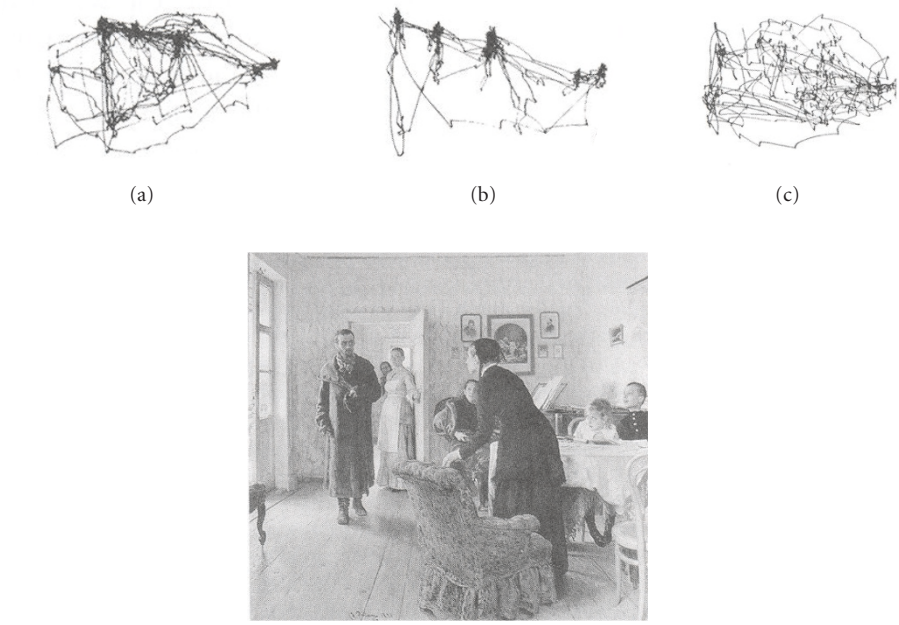
An artificial active vision system uses observer-controlled input sensors. Its main goal must be actively to extract the requested information in order to solve a given task. A wide literature proposes many systems built around the active vision paradigm. The majority of these systems have been driven by the "robotic" approach and are based on a robotic head. A large survey up to 1996 can be found in [11, 12].

Another trend considers algorithmic aspects and focuses on gaze control using foveated sensors with a log-polar mapping. This method can be applied at the sensor level (imager), at the image processing level or both. At the sensor level, some dedicated imagers based on logarithmic-structured space-variant pixel geometry have been implemented. The main advantage of these methods is the ability to cover wide work spaces with high acuity and a small number of pixels. Several descriptions of the advantages of using space-variant, or logmap, architectures for video sensors have been proposed [13–15]. Another logmap device consists of an emulated sensor based on a conventional CCD and an image warp algorithm embedded on a microcontroller [16]. More recently, a new trend towards smaller active vision systems comparable in size to the human head is pushing the limit of motor, gearbox, and camera design [17–19].

However, as mentioned above, most work dedicated to active vision systems is concentrated in the robotic field. In contrast, the main motivation for the work presented here is to propose a system truly resulting from the human visual system. Consequently, our approach needs a dedicated architecture for which the FPGA proves to be essential. This architecture is presented and discussed in the next section.

3. ARCHITECTURAL FEATURES

The main purpose of our architecture is to allow the implementation of early vision processes as in the human or primate visual system. In these systems, it is well known that the first neural layers (in the retina) prefilter the visual data flow in order to select only the conspicuous information. From this prefiltered information, an attentional processing allows focusing on the selected target. In the literature, several computational models of visual attention can be found. The first representative model was proposed by Koch and Ullman in [20] and has been recently revised by Itti et al. [21]. In these models, the purpose of the saliency map is to combine the "salient" or "conspicuous" location information from each of the lower feature maps into a global measure to determine how different a given location is from its surroundings. This technique is used to guide selective attention. The design of our active vision system is based on this kind of approach where we assume that the strategy of visual processes can be divided into the following three successive tasks.



(a) (b) (c)

Figure 1: Saccadic eye movements and task-driven strategy: examples of eye-scanning records obtained by Yarbus [7]. Observers were given different instructions while viewing the picture "They did not expect him" by Ilya Repin. Each of the traces shows a three-minute record of eye scanning with the following instructions: (a) free examination, (b) following request to give the ages of the people, (c) remember the position of the people and objects in the room.

Attention

This is the initializing step of the process. Whole images are grabbed while waiting for the building of the saliency maps. These maps are built in parallel and represent/code conspicuity within the visual field along particular dimensions (e.g., color, orientation, or motion). The result of this step is a set of ROIs (Regions Of Interest).

Focusing

This step allows the generation of the geometry of an ROI (rectangular, tilted, foveal, circular, ...) and the optimization of the signal/noise ratio: contrast optimization in an ROI [22], tracking of an ROI in motion, and so forth.

High-level processing

This last step comprises different kinds of tasks such as identification and classification.

The attention stage needs strong parallelization, on the one hand to respect real-time constraints, and on the other hand because of the intrinsic characteristics of the algorithms. As examples, some classical algorithms in an attention task used to build an efficient saliency map are motion detection, Gabor filters, and color segmentation. However, the characteristics of particular visual tasks may require dedicated image processing and only an FPGA approach allows such flexibility. For architectures such as these, a Stratix EP1S60 from Altera has been chosen. This choice is detailed below. The need for strong parallelization was what led us to connect 5×2 MB SRAM synchronous memory blocks. Each 2MB memory has private data and address buses. Consequently, in the FPGA, 5 attention processes (using 2 MB each) can address all the memory at the same time and an SDRAM module socket provides an extension of the memory to 64 MB (Figure 2).

The focusing stage must control the imaging devices in order to address only the ROI and to optimize the analog signal conversion. That is the reason why the sensing board has been designed around a CMOS imager and a set of 4 digital/analog converters. A set of inertial sensors has been added in order to estimate the movements of the camera and improve the perception (stabilization and depth estimation [23]).

In our approach, the high-level processing has to be performed on a host computer rather than on the embedded system. In order to send the data, the smart camera is connected via a high speed communication (USB 2.0 or FireWire).

The embedded system is integrated into a modular architecture consisting of three boards: the sensing board, the processing board, and the communication board. An overview of the smart camera is shown in Figure 3 and a structural description presents the stacked structure with 3 boards.

3.1. System on programmable chip features

As described in the previous section, the sensor was designed around a Stratix EP1S60 manufactured by Altera. This

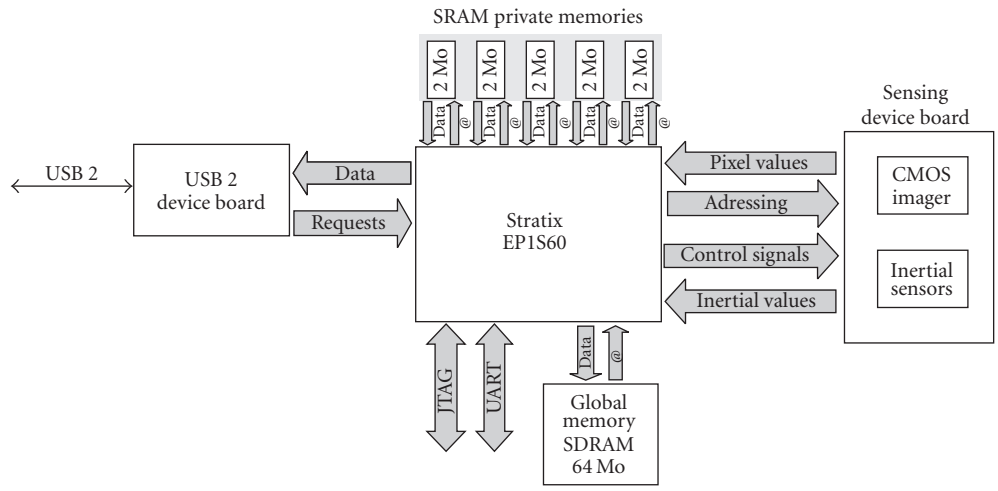


Figure 2: Architecture of the sensor.

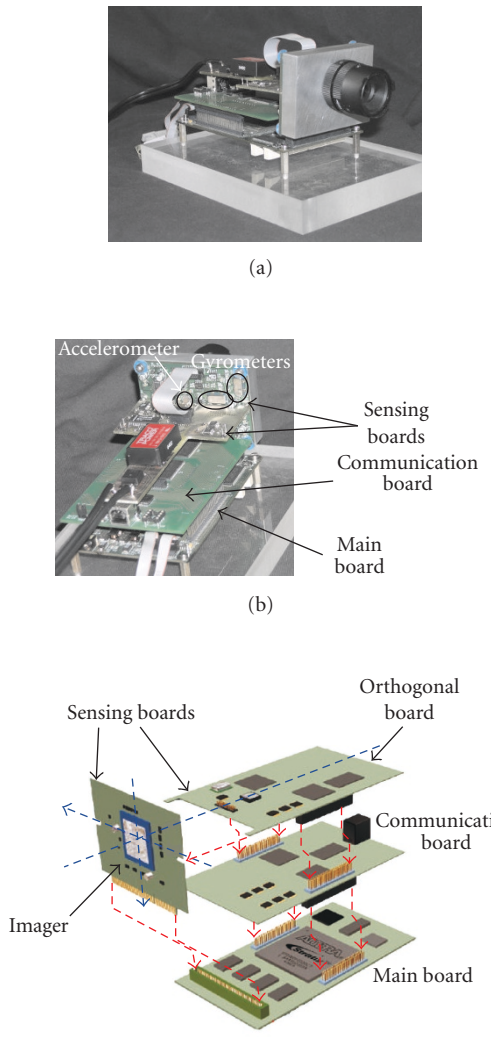


Figure 3: (a) Front view of the camera, (b) back view of the camera, (c) structural description.

component enables a high density of integration (57120 logic elements). It also has three further main advantages which guided our choice.

Firstly, the Stratix is optimized to maximize the performance benefits of SOPC integration based on an *NIOS* embedded processor. An *NIOS* processor is a user-configurable soft core processor, allowing many implementation and optimization options. The *NIOS* CPU is a pipelined general-purpose RISC microprocessor which supports both 32-bit and 16-bit architectural variants. Both 16- and 32-bit variants use 16-bit instructions. For our sensor, the main advantage of this soft core processor is its extensibility and adaptability. Indeed, users can incorporate custom logic directly into the *NIOS* arithmetic logic unit (ALU). Furthermore, thanks to a dedicated bus (Avalon bus), users can also connect into the SOPC on-chip processor and custom peripherals. They can thus define their own instructions and processor peripherals to optimize the system for a specific application.

Secondly, the Stratix integrates *DSP Blocks*. These embedded *DSP Blocks* have been optimized to implement several DSP functions with maximum performance and minimum logic resource utilization. Each DSP block offers multipliers, adders, subtractors accumulators, and a summation unit functions that are frequently required in typical DSP algorithms. Each DSP block can also support a variety of multiplier bit sizes (9×9, 18×18, 36×36) and operation modes (multiplication, complex multiplication, multiply-accumulation and multiply-addition) and can offer a DSP throughput of 2.8 GMACS per DSP block. The EP1S160 device has 18 *DSP Blocks* that can support up to 144 9×9 multipliers. These embedded *DSP Blocks* can also be used to create DSP algorithms and complex math routines in high-performance hardware. These can then be accessed as regular software routines or implemented as custom instructions on the NIOS CPU. For example, a cumbersome algorithm can be implemented in hardware and directly executed in software using a custom instruction. This gives designers the

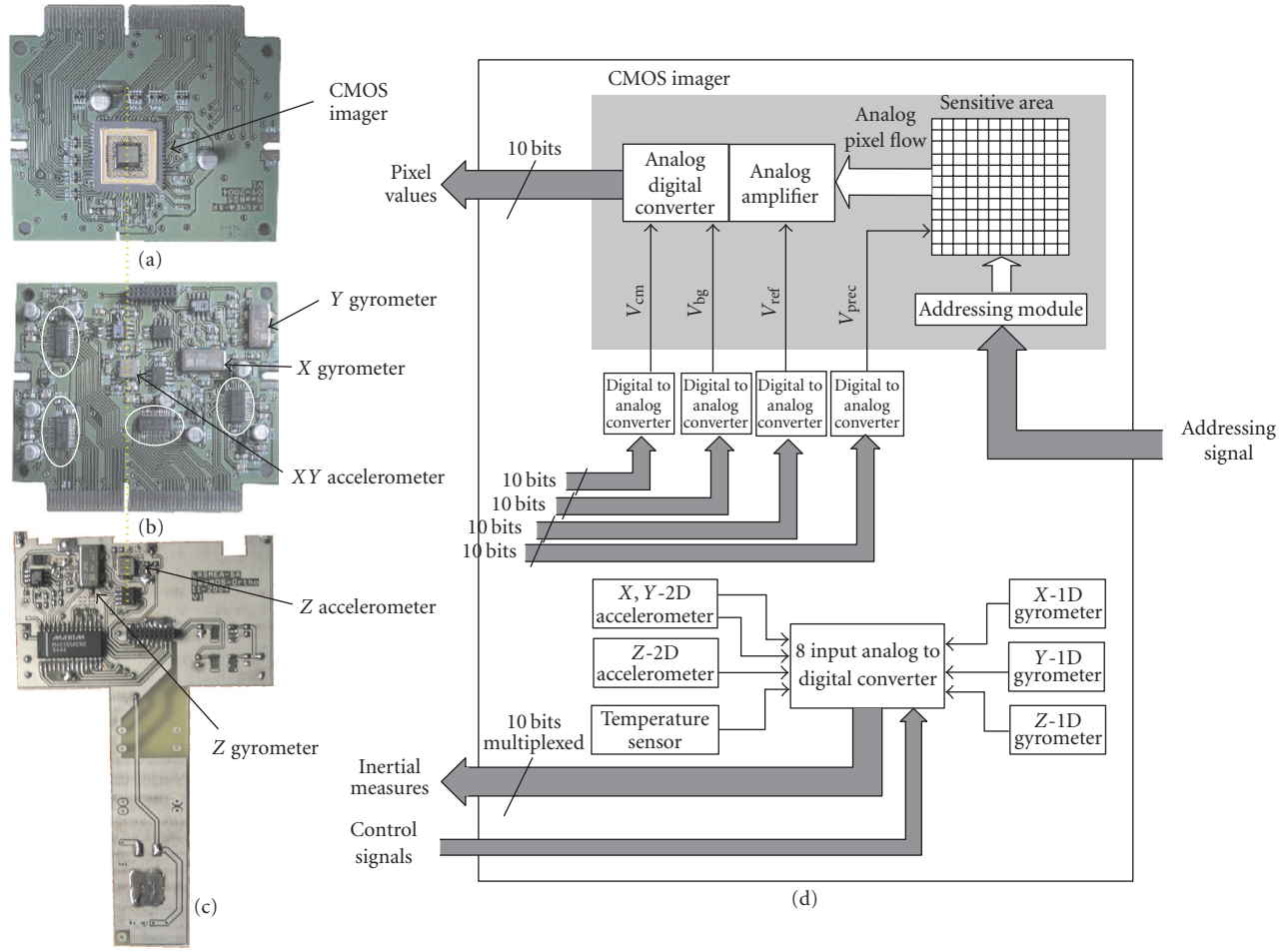


FIGURE 4: Sensing board: (a) front view; (b) back view; (the white circles show the 4 digital-analog converters); (c) orthogonal board; (d) global synoptic of the sensing boards.

flexibility and portability of high-level software design, while maintaining the performance benefits of parallel hardware operations in FPGAs.

Lastly, the Stratix device incorporates a configurable internal memory called TriMatrix memory. The TriMatrix memory is composed of three sizes of embedded RAM blocks. The Stratix EP1S60 TriMatrix memory includes 574 M512 blocks (32 × 18 bits), 292 M4K blocks (128×36 bits), and 6 M-RAM blocks (4 K × 144 bits). Each of these blocks can be configured to support a wide range of features and to synthesize a wide variety of RAM (FIFO, double ports). With up to 5 Mbits of fast RAM, the TriMatrix memory structure is therefore appropriate for handling the bottlenecks arising in sensor vision algorithms.

3.2. *Sensing device*

This module consists of a CMOS imager manufactured by Neuricam, two 2D accelerometers from analog devices and three 1D gyrometers from Murata. The imager allows full 2D addressing with a column bus and a row bus. It has a resolution of 640 × 480 (VGA) and provides a broad dynamic range (120 db) due to the logarithmic response of its pixel structure. Four digital-to-analog converters allow modification of the four analog voltages of the imager: analog signal offset, digital conversion range, voltage reference, and a pixel precharge voltage. These four converters are used to optimize the conversion range. In effect, the CMOS imager has a logarithmic curve that enables the broad dynamic range (120 dB).

The inertial set is composed of two 2D linear accelerometers *ADXL311* designed by analog devices and three gyrometers *ENC03 - M* designed by Murata. These sensors are soldered onto the imager PCB and aligned with the imager axis. A single 8-input analog-to-digital converter allows conversion of the different axis measurements. It is important to notice that a temperature sensor is included in this board to regulate the inertial sensors' deviations (see Figure 4).

4. ARCHITECTURAL DESIGN

The major difference between biological and artificial vision systems most probably lies in their flexibility. In order to develop adaptive capacities, the hardware architecture

previously described is designed to implement some specific low-level processing dedicated to early vision. This low-level processing attempts to establish an efficient interface between sensitive elements and high-level perception systems. As explained in Section 3, our strategy for efficient visual perception is based on three layers: attention, focusing, and high-level processing. This approach adopts a pyramidal method which reduces the amount of data flow. Typically, we can consider a simple system built around an attentional module based only on color segmentation and a focusing module based on template tracking (Figure 5). In the following, a detailed description of the FPGA organization is presented. All the "standard modules" (FPN correction, addressing module) are described and designs for focusing and attentional modules are proposed.

4.1. Implementation approach

The implementation of such an approach requires the management, sequentially and concurrently, of the execution of the routines previously described. Indeed, all task-oriented execution (attention, focusing, and identification) is controlled by supplied results and these three layers possibly have to share areas of interest. Moreover, the information bottleneck located in the imager level should be continuously optimized to ensure high performance. In our hardware architecture, these functions are carried out by what we term a "Sequencer" (Figure 6-M0) and are performed on the NIOS soft-core processor. This solution has two main advantages. Firstly, we benefit from software flexibility to define the routines' interactions; and secondly, the soft-core processor allows an efficient architectural matching with the other parts of the supervision unit.

An internal RAM (Figure 6-R0) is used to store the instruction sequences which define the sequencer behavior according to the task under consideration. The host computer which uses our embedded system communicates with it through a standard communication bus (USB 2.0 protocol) and sends requests in order to indicate to the sequencer the relevant behavior to adopt. More precisely, according to the controls (and potentially a set of parameters) passed in a dedicated stack, the sequencer chooses preestablished interactions between the modules (Figure 6-P4) which constitute a dedicated processing chain. This architectural module implements the previously described routines of environmental adaptation, attention, focusing, and low-level identification (Figure 6-P6). A number of these modules, due to environmental adaptation (processing No. 1 to i), modify the pixel flow which is going to be used by the attention, focusing, and identification modules (processing No. j to $j + 1$). The different data flows (corrected windows of interest and inertial measurements) can be used by these modules to perform computing. The set of results that are provided by these processing modules are collected in a buffer. This is how the sequencer selects results to send to the host computer. The sequencer is going to use a part of these results to perform visual feedback on sensing devices (Figure 6-S0) using dedicated control modules (Figure 6-P0, P1 and P3). We note

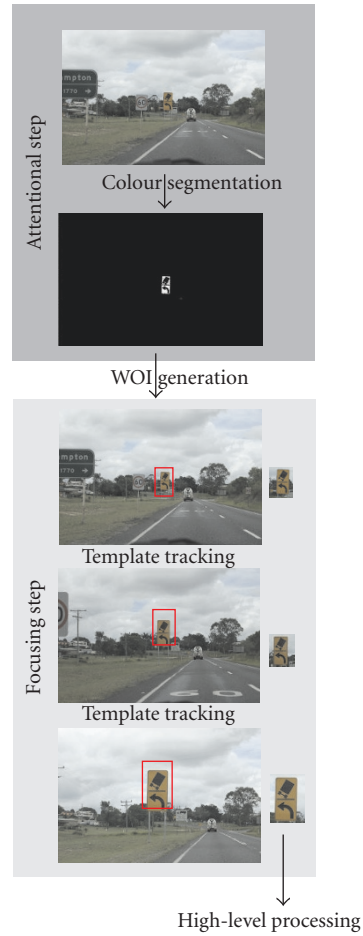


Figure 5: Example of perception strategy.

in Figure 6 the module P2 which works the external RAM R1. This module performs the fixed pattern noise correction which is absolutely essential with the image sensor technology we use (described in Section 4.3). Lastly, the dedicated communication module P5 is a multiplexer that synchronizes the corrected pixels flows and the sequencer results flows for sending to the host computer.

The sequencer constitutes an active interface between the sensing device, processing chain, and the host computer. The modular processing chain is synchronized with raw pixel flow control provided by the CMOS imager. Finally, this control allows dynamic control of the global sensor state according to global visual data coherence.

4.2. Addressing module (Figure 6-P0)

The goal of the *address generator device* is to compute line and column addresses of the current window of interest. The shape of the window is actually rectangular and is set by 5 parameters: position (X, Y), size (H, W), and orientation (α).

The address generation is based on the well-known "Bresenham" graphical algorithm [24]. For the computation of

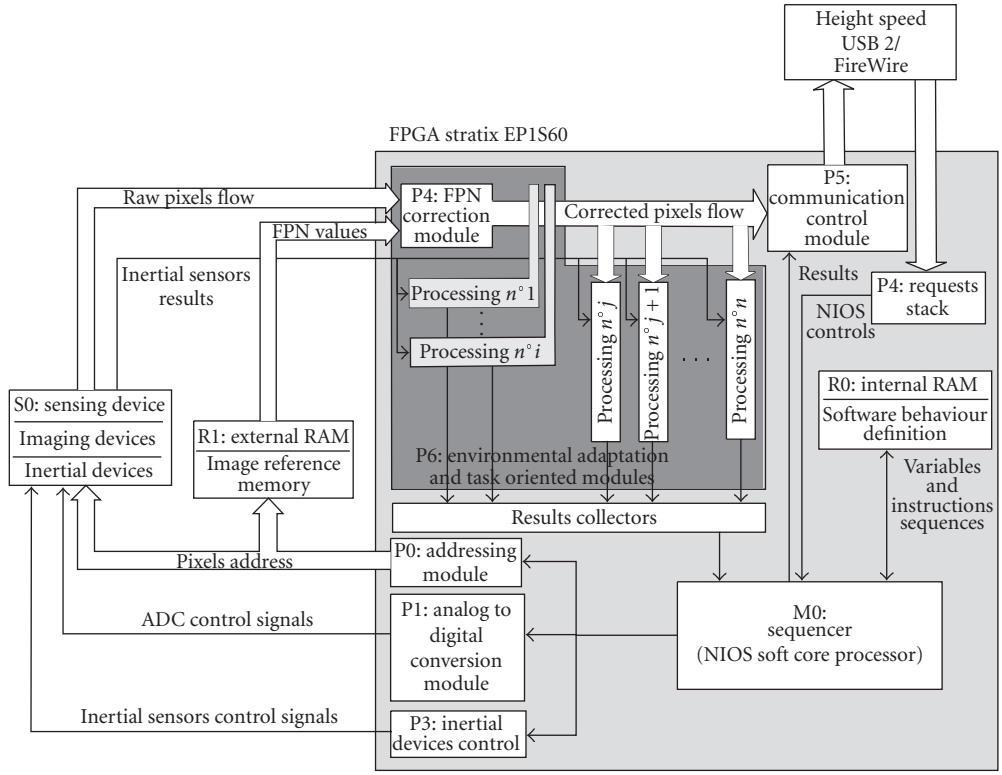


Figure 6: Block diagram of architecture adopted.

the tilted rectangular window addresses, we have implemented an architecture based on a recent method for drawing straight lines suitable for raster-scan displays and plotters developed by Bond [25].

The approach proposed by Bond is based on signal processing concepts related to resampling, multirate processing, and sample rate conversion. The x-coordinates of each pixel can be viewed as a uniform sample set, and the y-coordinates represent another sample set. Assuming the slope of the line is within the first octant, the y-coordinate is generated by resampling the set of x-coordinates. To control this resampling, the fractional part of the y-coordinate is stored in a control variable as an integer. The algorithm can be summarized by these few lines of code (see Algorithm 1).

$(X1, Y1)$ and $(X2, Y2)$ are the coordinates of the segment represented in Figure 7. n is the number of bits which are used to store the fractional part of the y-coordinates. Cvar is the fractional part of the y-coordinate. Incr is an integer variable used to store slope value. Carry is an overflow indicator for the operation Cvar+ = Incr;, and (X, Y) are the iterative coordinates of the line represented in Figure 7.

The extension of the algorithm to other octants is performed by interchanging the roles of x and y, and changing the signs of the coordinates x and y. The internal architecture of the *address generator device* is illustrated in Figure 8. The range of window tilt is encoded in a natural binary-coded variable named *Angle*. The first three MSB bits of this variable define the octant (Figure 8). The other bits of *Angle* are

```
(i) Initialization step
        Incr = ((Y2 − Y1)/(X2 − X1)) ∗ 2^n
        X = X1
        Y = Y1
        Cvar = 1/2 ∗ 2^n
(ii) Loop
        repeat {
                Cvar+ = Incr
                Y+ = Carry
                X++
               }
        until {X = X2}
```

Algorithm 1

used by two functions, based on the Bond algorithm, to generate the fixed-sample variable *FixCoor* and the resampled variable *ResCoord* for each window of interest dimension. According to the quadrant, a decoder defines the sign of the *FixCoor* and *ResCoord* variables. When the line is located in octants 2, 3, 6, or 7, the decoder causes the use of the complementary angle and the inversion of *FixCoor* and *ResCoord*. Finally, the sum of the line coordinates, column coordinates, and position vector of the window on the imager give the iterative X and Y address of each pixel. The implementation of

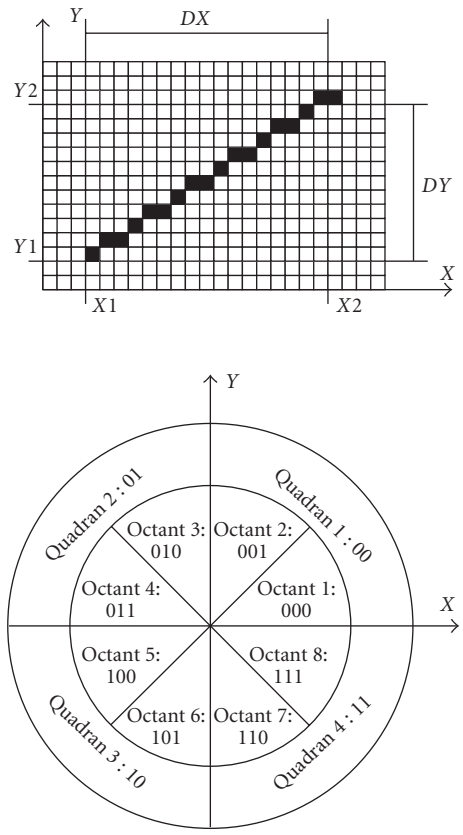


Figure 7: Sampled tilted line and octants encoding.

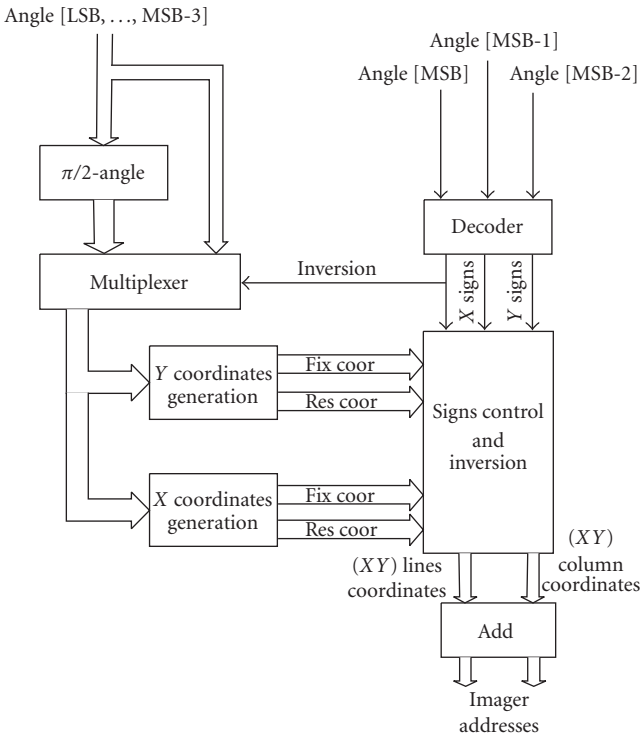


Figure 8: Synoptic diagram of addressing module.

Table 1

Total logic elements	150	< 1%
Total memory bits	0	0%
DSP block 9-bit elements	0	0%
System clock frequency	190 MHz	—

Table 2

Total logic elements	65	< 1%
Total memory bits	85	< 1%
DSP block 9-bit elements	0	0%
System clock frequency	241 MHz	—

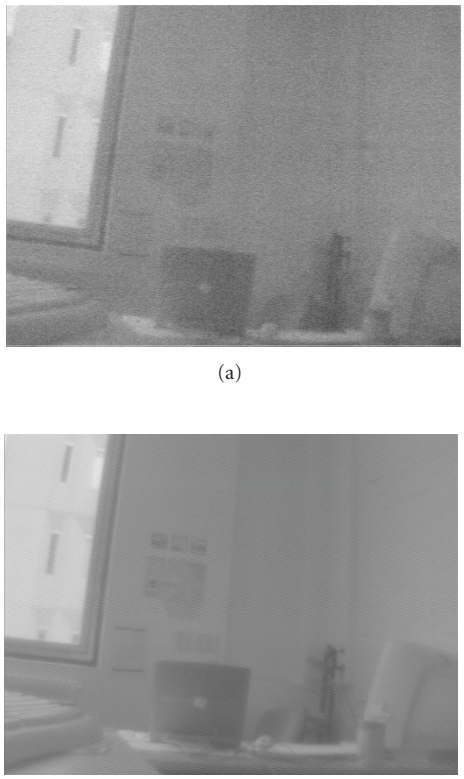
this module in the FPGA is characterized by the parameters shown in Table 1.

4.3. *Fixed pattern noise correction module (P4)*

Due to the technological limits, a classical CMOS imager (without an embedded CDS correction) provides a raw pixel flow with a high FPN. Indeed, the nonuniformity of the electrical characteristics of each pixel involves an additional stage of correction. In order to even out the electrical response of each pixel, the module called *FPN correction module* (Figure 6) subtracts the reference values (of the FPN) from the pixel flow. These referent values represent offset differences between each pixel for the same illumination (Figure 9). The set of these values constitutes a reference image. In order to carry out this correction, the sensor integrates a module in order to load it from an external RAM (Figure 6-R1). The implementation of this module in the FPGA is characterized by the parameters shown in Table 2.

5. EXAMPLE OF ATTENTION MODULE IMPLEMENTATION: MOTION DETECTION

Based on an image difference method (Figure 10), this algorithm looks for moving objects in a scene. In the image plan, motion is transduced to temporal and spatial gray-level changes. This module detects temporal changes and defines a rectangular window around the moving object. In the first step, a difference image is obtained through subtraction of images i and $i-1$. Then the difference image is thresholded, and its vertical projection is calculated (line-by-line pixel sum in each column). A peak detector is applied to the vertical projection, giving the horizontal position of the moving objects found in the scene. The horizontal projection inside each vertical zone detected previously is then calculated, and a second peak detection is applied to define the vertical position of the moving objects. In this way, it is possible to define the position of several moving objects in the image simultaneously. This information can be used as a parameter for another algorithm, such as a tracker.



(a)

(b)

Figure 9: Images (a) without and (b) with the subtraction of the image reference.

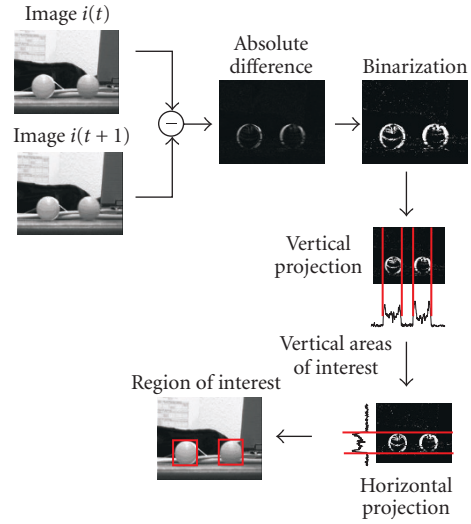


Figure 10: Motion detection processes.

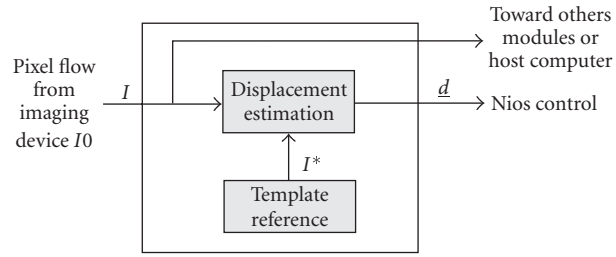


Figure 11: Template tracking module.

6. EXAMPLE OF FOCUSING MODULE: TEMPLATE TRACKING

As explained in Section 3, we assume that an active vision process can be split into three layers: attention, focusing and interpretation. The custom module proposed as an example in this section is a solution to the focusing layer. A design dedicated to an efficient template tracking implementation is presented. The main idea of template tracking is to estimate the displacement of a focusing window called W between time t and $t + \partial t$.

This module (Figure 11) comprises two parts: a memory to store the reference template denoted I^* and a dedicated architecture for the displacement estimation. The architecture adopted is based on a derivation of the Kanade-Lucas-Tomasi algorithm [26]. This algorithm is an iterative method to estimate displacement between two frames (I^* and I). The proposed method is based on the calculation of the dissimilarity between two images as follows:

$$\varepsilon = \int\int_W \left[I\left(\mathbf{x} + \frac{\mathbf{d}}{2}\right) - I^*\left(\mathbf{x} - \frac{\mathbf{d}}{2}\right)\right]^2 w(\mathbf{x})d\mathbf{x}, \quad (1)$$

where $\mathbf{x} = [x \;\; y]^T$, the displacement $\mathbf{d} = [d_x \;\; d_y]^T = \partial\mathbf{x}/\partial t$, and the weighting function $w(\mathbf{x})$ is usually set to the constant 1. To find the displacement $\mathbf{d}$, we set $\partial\varepsilon/\partial\mathbf{d}$ to zero. If we consider the Taylor series expansion of I and I^*, respectively, about $\mathbf{x} - \mathbf{d}/2$ and $\mathbf{x} + \mathbf{d}/2$, we obtain

$$\frac{\partial\varepsilon}{\partial\mathbf{d}} = \int\int_W \left[I(\mathbf{x}) - I^*(\mathbf{x}) + \frac{1}{2}\mathbf{g}^T(\mathbf{x})\mathbf{d}\right]\mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x} = 0, \quad (2)$$

where

$$\mathbf{g} = \begin{bmatrix} \dfrac{\partial}{\partial x}(I + I^*) \\ \dfrac{\partial}{\partial y}(I + I^*) \end{bmatrix}. \quad (3)$$

Finally, the displacement $\mathbf{d}$ can be estimated by solving the equation:

$$Z\mathbf{d} = \mathbf{e}, \quad (4)$$

where Z is the following 2×2 matrix:

$$Z = \int\int_W \mathbf{g}(\mathbf{x})\mathbf{g}^T(\mathbf{x})w(\mathbf{x})d\mathbf{x}, \quad (5)$$

and $\mathbf{e}$ is the following 2×1 vector:

$$\mathbf{e} = 2\int\int_W [I^*(\mathbf{x}) - I(\mathbf{x})]\mathbf{g}(\mathbf{x})w(\mathbf{x})d\mathbf{x}. \quad (6)$$

Due to the Taylor expansion, this algorithm is not exact and needs iterations to find the correct displacement.

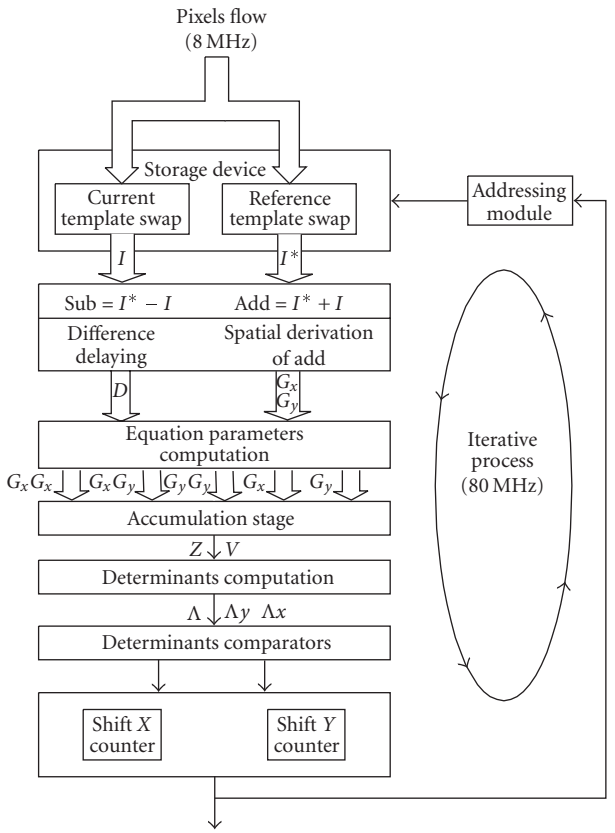


Figure 12: Implemented architecture.

The estimated displacement at each iteration i is denoted $\mathbf{s}_i = (s_x \;\; s_y)^T$ and is calculated by

$$\mathbf{s} = 2 \cdot Z^{-1} \cdot \mathbf{e}, \tag{7}$$

where

$$\mathbf{e} = \left(\sum_W D \cdot Gx \;\; \sum_W D \cdot Gy \right)^t,$$
$$Z = \begin{pmatrix} \sum_W Gx \cdot Gx & \sum_W Gx \cdot Gy \\ \sum_W Gx \cdot Gy & \sum_W Gy \cdot Gy \end{pmatrix}, \tag{8}$$

where $G_x = \partial(I + I^*)/\partial x$, $G_y = \partial(I + I^*)/\partial y$ and $D = (I^* - I)$ is the interframe difference.

The displacement $\mathbf{d} = (dx \;\; dy)^T$ is the result of the iterative process such that

$$\mathbf{d} = \sum_{i=1}^{i=N} \mathbf{s}_i, \tag{9}$$

where N is the maximum number of iterations allowed by the processing. In our case, because of NIOS control, the iterative process is limited to 80 MHz and the pixel flow runs at 8 MHz (corresponding to $N = 80/8 = 10$ iterations max).

The architecture developed to implement this algorithm is presented in Figure 12. The first module called "storage device" allows the storage of the reference frame I^* and swapping of the current pixel flow between two double-port memories. This module performs the two functions simultaneously in order to ensure the pixel flow rate. The displacement between the reference template I^* and current window I of interest is simultaneously computed during the storage of the next window of interest.

```
if ‖Λx‖ > ‖Λ‖/4
   then
   if sign(Λx) ⊕ sign(Λ)
      then CTx − −
      else CTx + +
and
if ‖Λy‖ > ‖Λ‖/4
   then
   if sign(Λy) ⊕ sign(Λ)
      then CTy − −
      else CTy + +
```

Algorithm 2

In the first step, the difference (sub) between the two images and the spatial derivatives of their sum (add) are computed and synchronized. The computation of spatial derivatives (Gx and Gy) is based on a set of FIFOs and multiplier-accumulators which apply a (3×3) convolution mask to the data flow. The convolution kernel mask is the Gaussian derivative function.

In the second step, Gx, Gy, and Sub_d are applied to a set of multipliers in order to compute the coefficients G_xG_x, G_xG_y, G_yG_y, G_xD, and G_xD. The accumulation of each allows computation of the elements of Z and $\mathbf{e}$.

The solution is obtained by the evaluation of the following three determinants:

$$\Lambda = \sum_W Gx \cdot Gx \cdot \sum_W Gy \cdot Gy \left(\sum_W Gx \cdot Gy \right)^2,$$
$$\Lambda_x = \sum_W D \cdot Gx \cdot \sum_W Gy \cdot Gy - \sum_W D \cdot Gy \cdot \sum_W Gx \cdot Gy,$$
$$\Lambda_y = \sum_W Gx \cdot Gx \sum_W D \cdot Gy - \sum_W Gx \cdot Gy \cdot \sum_W D \cdot Gx. \tag{10}$$

According to the signs and the comparison of Λ, Λ_x, and Λ_y, the displacement counters CT_x, CT_y are updated as shown in Algorithm 2.

The updating of the reference template position is carried out according to the counter values. This process is iteratively repeated and allows detection of the correct translation vector between the two frames. Lastly, the estimated translation vector is used to update the position of the window of interest in the CMOS imager in order to track the reference template.

The implementation of this architecture on a Stratix EP1S60 leads to the parameters shown in Table 3.

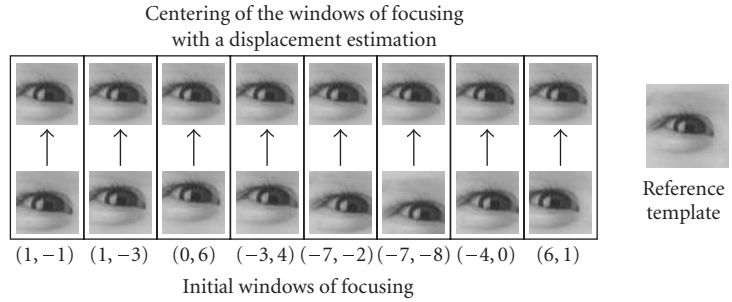


Figure 13: Results of the template tracking architecture. To evaluate the robustness of the approach, the image is artificially moved with a given displacement indicated under each image.

Table 3

Total logic elements	4238	7%
Total memory bits	546 816	10%
DSP block 9-bit elements	94	65%
System clock frequency of the design	140 MHz	—

To illustrate the algorithm, several images and simulation results are presented. This simulation was performed using ModelSim software with the VHDL description of our system.

7. CONCLUSION

The computation of low-level vision tasks in real time is the first and fundamental step in building an interactive vision system. This paper proposes an alternative to classical architectures with a highly versatile architecture dedicated to early image processing. The proposed embedded system attempts to define a global coordination between sensitive elements, low-level processing, and visual tasks.

The approach, based on FPGA technology and a CMOS imager, reduces the classical bottleneck between sensor and processing. The FPGA component ensures a high interaction rate between the CMOS imager and low-level processing. This interaction is used to select useful information earlier in the acquisition chain than for more traditional systems. It then focuses processing resources. This capacity is used to control the sensor state according to the visual task and the environment evolution. Our implementation of the FPGA and CMOS imager technologies results in high-speed vision, real-environment vision, and the efficient design of embedded systems. Among prospective algorithm candidates, we can cite the works performed on dynamically reconfigurable components such as the ARDOISE[1] project [27]. This evolution of FPGA technology seems to be attractive for performing dynamic control of the acquisition chain. Rather than having a control system state, the system itself can be physically changed and giving a higher level of suitability for many algorithms. We have also developed a DSP board in order to improve the computation capabilities of our system. With this device, our embedded system will evolve into a heterogeneous architecture, and new research into codesign between the FPGA and the DSP will be necessary.

Moreover, to test the validity of our approach, several visual tasks will be implemented. Our objective is to identify elementary functions in order to define a library of architectural modules. Of course, this library will provide efficient solutions to attention resolution, focusing, and identification of subtasks according to specific applications. Finally, we plan to work on the development of software tools to facilitate the implementation of complex vision tasks

[1] ARDOISE: architecture reconfigurable dynamiquement orientée image et signal embarquable.

REFERENCES

[1] A. DeHon, "Density advantage of configurable computing," *Computer*, vol. 33, no. 4, pp. 41–49, 2000.

[2] A. Benedetti and P. Perona, "Real-time 2-D feature detection on a reconfigurable computer," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pp. 586–593, Santa Barbara, Calif, USA, June 1998.

[3] J. Woodfill and B. Von Herzen, "Real-time stereo vision on the PARTS reconfigurable computer," in *Proceedings of 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines (FCCM '97)*, pp. 201–210, Napa Valley, Calif, USA, April 1997.

[4] D. Benitez and J. Cabrera, "Reactive computer vision system with reconfigurable architecture," in *Proceedings of 1st International Conference on Computer Vision Systems (ICVS '99)*, pp. 348–360, Las Palmas, Gran Canaria, Spain, January 1999.

[5] W. Böhm, J. Hammes, B. Draper, et al., "Mapping a single assignment programming language to reconfigurable systems," *Journal of Supercomputing*, vol. 21, no. 2, pp. 117–130, 2002.

[6] D. Marr, *Vision*, W. H. Freeman, San Francisco, Calif, USA, 1982.

[7] A. L. Yarbus, *Eye Movements and Vision*, Plenum Press, New York, NY, USA, 1967.

[8] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," in *Proceedings of 1st International Conference on Computer Vision*, pp. 35–54, London, UK, June 1987.

[9] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 996–1005, 1988.

[10] D. H. Ballard, "Animate vision," *Artificial Intelligence*, vol. 48, no. 1, pp. 57–86, 1991.
[11] C. S. Andersen, *A framework for control of a camera head*, Ph.D. thesis, Laboratory of image analysis, Aalborg Universtity, Aalborg, Denmark, 1996.
[12] T. Vieville, *A Few Steps Towards 3D Active Vision*, vol. 33 of *Springer Series in Information Sciences*, Springer, New York, NY, USA, 1997.
[13] J. Van der Spiegel, G. Kreider, C. Claeys, et al., "A foveated retina-like sensor based on CCD technology," in *Analog VLSI Implementation of Neural Systems*, C. Mead and M. Ismail, Eds., pp. 189–210, Kluwer Academic, Boston, Mass, USA, 1989.
[14] A. S. Rojer and E. L. Schwartz, "Design considerations for a space-variant visual sensor with complex-logarithmic geometry," in *Proceedings of the 10th International Conference on Pattern Recognition*, vol. 2, pp. 278–285, Atlantic City, NJ, USA, June 1990.
[15] M. Tistarelli and G. Sandini, "On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 401–410, 1993.
[16] B. B. Bederson, *A miniature space-variant active vision system: cortex-I*, Ph.D. thesis, New York University, New York, NY, USA, 1992.
[17] Y. Kuniyoshi, N. Kita, S. Rougeaux, and T. Suehiro, "Active stereo vision system with foveated wide angle lenses," in *Proceedings of 2nd Asian Conference on Computer Vision (ACCV '95)*, pp. 191–200, Singapore, December 1995.
[18] P. M. Sharkey, D. W. Murray, and J. J. Heuring, "On the kinematics of robot heads," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 437–442, 1997.
[19] S. Truong, J. Kieffer, and A. Zelinsky, "A cable-driven pan-tilt mechanism for active vision," in *Proceedings of Australian Conference on Robotics and Automation (ACRA '99)*, pp. 172–177, Brisbane, Australia, March-April 1999.
[20] C. Koch and S. Ullman, "Shifts in selective visual attention: towards the underlying neural circuitry," *Human Neurobiology*, vol. 4, no. 4, pp. 219–227, 1985.
[21] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.
[22] P. Chalimbaud and F. Berry, "Contrast optimization in a multi-windowing image processing architecture," in *Proceedings of IAPR Conference on Machine Vision Applications (MVA '05)*, Tsukuba Science City, Japan, May 2005.
[23] P. Chalimbaud, F. Berry, F. Marmoiton, and S. Alizon, "Design of a hybrid visuo-inertial smart sensor," in *Proceedings of IEEE International Conference on Robotics and Automation Workshop on Integration of Vision and Inertial Sensors (InerVis '05)*, Barcelona, Spain, April 2005.
[24] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.
[25] C. Bond, "A New Line Drawing Algorithm: Based on Sample Rate Conversion," 2002, http://www.crbond.com/.
[26] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, Pittsburgh, Pa, USA, April 1991.
[27] D. Demigny, N. Boudouani, R. Bourguiba, and L. Kessal, "Vers une méthodologie pour la programmation des architectures à reconfiguration dynamique," in *Actes du Workshop Adéquation Algorithmes Architectures En Traitement Du Signal et de L'image*, pp. 15–20, Janvier 2000.